멀티미디어 스트리밍 서버를 위한 인기도 기반 인터벌 캐싱의 블록 수준 세분화 기법

(Block Level Refinement of Popularity-Aware Interval Caching for Multimedia Streaming Servers)

권 오 훈 † 김 태 석 †† 반 효 경 ††† 고 건 ††††
(Ohhoon Kwon) (Taeseok Kim) (Hyokyung Bahn) (Kern Koh)

요 약 최근 VOD(Video-On-Demand) 서비스가 널리 이용되면서 멀티미디어 스트리밍 서버를 위한 데이타 캐싱 기법의 중요성이 점점 증가하고 있다. 기존 연구를 통해 인터벌 캐싱 기법과 이를 객체의 인기도를 반영하도록 확장한 기법들이 다양한 환경에서 우수한 성능을 나타냄이 입증되었다. 본 논문에서는 이와 같은 기존의 기법을 블록 수준으로 세분화할 경우 멀티미디어 스트리밍 서버의 성능을 더욱 향상 시킬 수 있음을 보인다. 실제 VOD 서버의 트레이스를 이용한 시뮬레이션 실험을 통해 본 논문이 제안한 알고리즘이 캐쉬 적중률과 스트림의 초기 지연 시간을 향상시킴을 보인다.

키워드 : 캐싱, 멀티미디어, 스트리밍, VOD(Video-on-Demand)

Abstract With recent proliferation of video-on-demand services, caching in a multimedia streaming server is becoming increasingly important. Previous studies have shown that request interval based caching and its extension for considering different video popularity performs well in various streaming environments. In this paper, we show that block level refinement of this existing scheme can further improve the performance of streaming servers. Trace driven simulations with real world VOD traces have shown that the proposed scheme improves the cache hit rate and the startup latency.

Key words : caching, multimedia, streaming, VOD (Video-on-Demand)

1. 서 론

멀티미디어 스트리밍 서버에서의 캐싱 기법은 서버 시스템의 성능 향상과 사용자의 서비스 대기 시간을 줄이는 효과적인 방법이다. 멀티미디어 객체는 크기가 크고 순차적으로 참조되므로 LRU(Least Recently Used) 알고리즘과 같은 전통적인 버퍼 캐쉬 관리 기법을 멀티미디어 서버에 그대로 적용하는 것은 효율적이지 못하다. 이러한 문제를 해결하기 위해 기존에 많은 연구들이 수행되었고, 그 대표적인 버퍼 관리 기법이 스트리밍 서비스의 요청 간격에 기반한 인터벌 캐싱(Interval Caching) 기법이다[1-6]. 인터벌 캐싱 기법은 동일한 객체에 대한 연속적인 요청(request)들 사이의 간격을 인터벌이라고 정의하고, 인터벌 내에 있는 데이타만 캐싱함으로써 적은 버퍼 공간을 사용하여 뒤따라오는 요청에게 디스크 입출력 작업 없이 메모리에서 직접 서비스해 주는 방법이다. 하지만, 이와 같은 인터벌 캐싱 기법은 멀티미디어 스트림의 길이가 충분히 길지 않거나 스트림 요청 간의 시간 차가 클 경우 캐싱의 효과를 거두기 어렵다. Kim 등은 인터벌 캐싱 기법을 확장하여 멀티미디어 객체의 인기도를 고려하는 PIC(Popularity-aware Interval Caching) 기법을 제안하였다[7,8]. PIC 기법은 각 멀티미디어 객체에 대한 과거의 요청 및 이들 사이의 인터벌에 근거해서 객체의 인기도를 추정하고, 추정된 인기도를 바탕으로 미래의 요청 시간을 예측하여 가상의 인터벌을 생성한다. 이를 통해 인기 있는 데이타의 경우, 요청이 있기 전에 가상 인터벌 내에 있는 데이타를 미리 캐싱하여 데이타가 실제 요청되었을 때 메모리에서 직접 서비스될 수 있도록 한다. 하지

† 학생회원 : 서울대학교 컴퓨터공학과
ohkwon@oslab.snu.ac.kr
†† 비 회 원 : 서울대학교 컴퓨터공학과
tskim@oslab.snu.ac.kr
††† 종신회원 : 이화여자대학교 컴퓨터공학과 교수
bahn@ewha.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터 공학과 교수
kernkoh@oslab.snu.ac.kr
논문접수 : 2006년 10월 24일
심사완료 : 2007년 1월 2일

만, PIC 기법에서는 시간이 흐르면서 가상 인터벌이 점차 멀티미디어 객체의 뒷부분으로 이동하여 초기 지연 시간(startup latency)을 효과적으로 줄일 수 있는 객체의 시작 부분을 캐슁하지 못한다. 이에 본 논문에서는 PIC 기법을 블록 단위로 세분화하여 이러한 문제점을 해결하고 나아가 멀티미디어 스트리밍 서버의 성능을 향상 시킬 수 있는 인기도 기반 인터벌 캐슁의 블록 수준 세분화(Block Level Refinement of Popularity-Aware Interval Caching, B-PIC) 기법을 제안한다. 실제 VOD 트레이스를 사용한 시뮬레이션을 통해서 본 논문이 제안한 알고리즘이 캐쉬 적중률(hit ratio)과 시작 블록 미스(start block misses)에 대해서 PIC 기법과 인터벌 캐슁, LRU(Least Recently Used), MRU(Most Recently Used) 알고리즘보다 우수한 성능을 나타냄을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 멀티미디어 스트리밍 서버에 대한 기존 캐슁 기법들에 대해 간단히 살펴본다. 3장에서는 멀티미디어 스트리밍 서버의 시스템 구조와 본 논문에서 제안하는 인기도 기반 인터벌 캐슁의 블록 수준 세분화 기법에 대해 설명한다. 4장에서는 제안된 캐슁 기법과 기존의 캐슁 기법과의 성능을 비교하고 그 결과에 대해 평가한다. 마지막으로, 5장에서 본 논문의 결론을 기술한다.

2. 관련 연구

본 연구와 같이 멀티미디어 스트리밍 서버를 위한 버퍼 캐쉬 관리 기법에 대한 다양한 연구들이 지속적으로 수행되고 있다. Dan과 Sitaram은 동일한 멀티미디어 객체에 대한 연속적인 요청들 사이의 오프셋(offset) 거리를 인터벌이라 정의하고, 인터벌 내에 있는 데이타만을 캐슁함으로써 디스크 입출력 작업 없이 뒤따라 오는 요청을 버퍼 캐쉬에서 직접 서비스 해주는 인터벌 캐슁 기법을 제안하였다[1,2]. 인터벌 캐슁 기법은 모든 연속적인 요청들의 쌍(pair)을 메모리 요구량의 순으로 정렬한다. 그런 다음 메모리 요구량이 적은 쌍을 우선적으로 캐슁한다. 이 방법은 가능한 많은 요청들을 동시에 메모리 버퍼에서 서비스하는 데에 목적이 있다. 즉 어떤 인터벌이 캐슁될 때, 해당 인터벌에 속한 후행 스트림 요청은 메모리 내의 버퍼 캐쉬에서 직접 서비스 받을 수 있다. Ozden 등은 인터벌 캐슁과 비슷한 개념을 지닌 거리 캐슁(distance caching)이라는 이름의 버퍼 캐쉬 관리 기법을 제안하였다[5,6]. 거리 캐슁 기법에서는 선행 스트림 요청까지의 오프셋 거리에 기반한 우선 순위를 각각의 스트림 요청에 할당한다. 그런 다음 모든 스트림 요청 중 가장 우선 순위가 낮은 요청(선행 요청으로부터의 오프셋 거리가 가장 먼 요청)이 사용하는 블록을 캐쉬에서 쫓겨날 블록으로 선택한다. 하지만, 인터벌 캐슁과 거리 캐슁 기법은 짧은 기간 내의 두 연속적인 요청에 대한 시간 지역성(temporal locality)만을 고려하고 멀티미디어 객체의 서로 다른 인기도를 고려하지 않는다. 그 결과 멀티미디어 객체의 크기가 충분히 크지 않거나 스트림 요청 간의 인터벌이 매우 긴 경우 캐슁의 효과를 얻을 수 없다. 이러한 문제점을 해결하기 위해 Kim 등은 인기도 기반의 인터벌 캐슁(Popularity-aware Interval Caching, PIC) 기법을 제안하였다[7,8]. 이 기법은 우선 각각의 멀티미디어 객체에 대한 과거의 요청 간격을 통해 미래에 요청될 시간을 예측한다. 그리고, 멀티미디어 객체에 대한 실제 요청 시간들 간의 인터벌 뿐 아니라 실제 요청 시간과 예측된 요청 시간 간의 인터벌을 캐슁의 대상으로 함께 고려한다. 이는 인기 있는 멀티미디어 객체의 경우 요청이 들어오기 전에 앞부분이 미리 캐슁되도록 하여 인터벌 캐슁의 약점을 효율적으로 보완한다.

최근 들어서, 다양한 환경에서 인터벌 캐슁 기법을 확장한 연구가 수행되고 있다. Sarhan과 Das는 NAD (Network Attached Disk) 계층 구조에서 인터벌 캐슁 기법을 확장하였고[3], Almeida 등은 멀티미디어 스트리밍 서비스를 위한 프록시 서버 계층에서 버퍼 캐쉬 단(buffer cache layer)에서는 인터벌 캐슁 기법을 사용하고, 디스크 캐쉬 단(disk cache layer)에서는 LFU (Least Frequently Used)기법을 사용하는 2단계 캐슁 기법을 제안하였다[4]. Lee 등은 멀티미디어 시스템에서 디스크 대역폭(bandwidth)을 효과적으로 사용하는 동시에 스트리밍 서비스가 일시적으로 끊어지는 현상(hiccup)을 없애는데 캐쉬 적중률만 고려하는 것은 적절하지 않음을 설명하고, PSIC(Preemptive but Safe Interval Caching)라는 이름의 새로운 버퍼 캐쉬 관리 기법을 제안하였다[9]. 더욱 최근에는 Fernandez 등이 ISC(Iteration Set Caching) 기법을 제시하였다[10]. 이 기법은 가변 비트율(variable bit-rate) 스트림 환경에서 우수한 성능을 얻기 위해 인터벌 캐슁 기법을 확장시킨 기법으로 기존의 인터벌 캐슁 기법이 인터벌 간의 우선 순위를 정적으로 결정하는 것과 달리 캐슁된 블록 간의 우선 순위를 동적으로 변할 수 있도록 하였다.

또한, Sen 등은 인기 있는 멀티미디어 객체의 시작부분을 멀티미디어 프록시(proxy) 서버에 저장함으로써, 사용자 지연시간의 단축과 스트리밍 서비스의 성능을 향상시키는 기법을 제안하였다[11]. Lim 등은 프록시 서버에서 멀티미디어 객체의 시작부분을 캐슁하는 데 있어서 멀티미디어 객체의 인기도에 따라 캐슁하는 시작부분의 크기를 달리하는 기법을 제안하였다[12].

3. 인기도 기반 인터벌 캐슁의 블록 수준 세분화 기법

본 장에서는 멀티미디어 스트리밍 서버를 위한 새로운 캐슁 기법을 제안한다. 3.1절에서는 멀티미디어 스트리밍 서버의 시스템 구조를 간단히 설명하고 3.2절에서는 본 논문이 새롭게 제안하는 블록 수준의 인기도 기반 인터벌 캐슁 기법에 대해 기술한다. 그리고 3.3절에서는 본 논문에서 제안하는 캐슁 기법의 효율적인 구현 방법에 대해 설명한다.

3.1 멀티미디어 스트리밍 시스템의 구조

멀티미디어 스트리밍 서버의 시스템 구조(system architecture)는 그림 1과 같이 디스크 입출력 관리자(disk I/O manager), 버퍼 관리자(buffer manager), 네트워크 관리자(network manager)로 구성된다. 버퍼 관리자는 시스템 내의 메모리 버퍼를 캐쉬(cache)와 미리 읽기 버퍼(read-ahead buffer)로 구분한다. 캐쉬는 디스크로부터 읽어 들인 데이타를 시간/공간 지역성을 고려해서 일정 기간 동안 메모리 내에 저장할 수 있는 공간을 말한다. 추후 동일한 데이타에 대한 입출력 요청이 있을 시, 디스크 입출력 없이 캐쉬 영역에서 서비스해줌으로써 시스템의 성능을 향상 시킬 수 있다. 또한, 디스크 입출력 작업은 상대적으로 큰 비용을 가지기 때문에, 입출력 작업 시 요청된 데이타만 읽어오는 게 아니라 그와 관련성이 있는 데이타도 미리 읽어 들인다. 이것을 미리 읽기 기법이라 하고, 미리 읽기 기법을 통해 읽어온 데이타를 저장하는 공간을 미리 읽기 버퍼라고 한다.

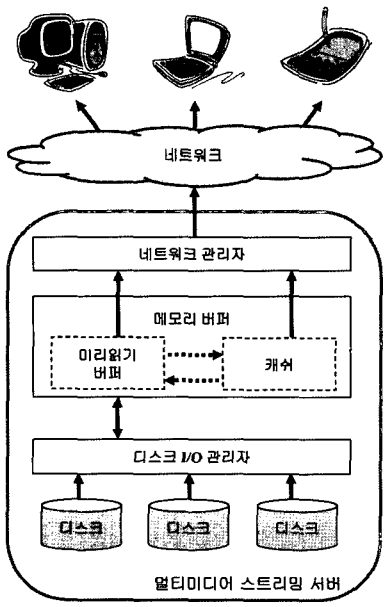


그림 1 멀티미디어 스트리밍 서버의 구조

캐쉬나 미리 읽기 버퍼에 저장된 데이타는 메모리 버퍼 내에 존재하기 때문에 실제로 캐쉬와 미리 읽기 버퍼 간의 물리적인 데이타 이동은 없으며 플래그를 사용해서 논리적인 구분만을 한다. 디스크 입출력 관리자는 요청된 블록이 메모리 버퍼에 없을 경우 빈 블록을 얻고 디스크 입출력을 실행한다. 마지막으로 네트워크 관리자는 메모리 버퍼로부터 필요한 데이타 블록을 읽어 네트워크를 통해 클라이언트에게 전송한다.

3.2 인기도 기반 인터벌 캐슁의 블록 수준 세분화 기법

인터벌 캐슁 기법에서는 동일한 멀티미디어 객체에 대한 연속적인 요청을 선행 요청과 후행 요청으로 구분한다. 그런 다음 후행 요청은 선행 요청이 읽어 들인 데이타가 메모리 버퍼에 존재하는 경우 디스크 입출력 없이 필요한 데이타를 버퍼에서 곧바로 서비스한다. 이와 같은 선행 요청과 후행 요청의 관계를 통해 소량의 메모리 공간만으로 후행 요청의 연속적인 캐쉬 적중을 가능하게 할 수 있다. 인터벌 캐슁 기법에서는 동일한 객체에 대한 연속적인 요청들 사이의 오프셋 거리를 인터벌이라 정의하고, 최대한 많은 수의 스트리밍 요청을 메모리 버퍼에서 동시에 서비스하는데 그 목적이 있다. 그러므로 한정된 버퍼 공간에 대해 인터벌 캐슁은 인터벌들을 메모리 버퍼 요구량 순으로 정렬하고, 버퍼 요구량이 적은 인터벌부터 우선적으로 캐슁한다.

Kim 등은 멀티미디어 객체의 인기도를 고려하기 위해 인터벌 캐슁 기법에 가상 인터벌(virtual interval) 개념을 접목시켰다[7]. 이 기법에서는 멀티미디어 객체에 대한 과거의 요청을 분석하여 언제쯤 새로운 요청이 들어올지를 예측한 후 가장 최근의 요청과 예측된 요청 간의 시간 간격을 가상 인터벌이라 정의한다. 이 때, 사용자의 실제 요청이 아니라 예측된 요청을 가상 요청(virtual request)이라고 명명한다[7,8].

본 논문이 제안하는 B-PIC 기법에서는 PIC 기법과 마찬가지로 과거 참조 패턴을 분석해서 멀티미디어 객체의 미래 참조 시점을 예측한다. 하지만, PIC 기법과는 달리 인터벌을 단위로 캐슁 여부를 결정하는 것이 아니라 인터벌(가상 인터벌을 포함)을 구성하는 각각의 블록 단위로 캐슁 대상을 결정한다. 이 때, 각 블록에 대한 캐슁 여부는 각 블록을 캐슁했을 때 얻을 수 있는 잠재적 이득을 계산하는 기대 참조 확률을 통해 결정한다.

수식 (1)은 과거 참조 패턴을 바탕으로 가장 최근의 실제 요청과 미래의 잠재적 요청 간의 간격을 예측하는 식이다. 식에서 I는 가장 최근의 실제 인터벌이고 PI_{k-1}은 $(k-1)$번째 가상 인터벌이다. 이 때, k 번째 가상 인터벌은 다음과 같이 계산한다.

$$PI_k = \alpha I + (1-\alpha)\, PI_{k-1} \quad (k > 1) \qquad (1)$$

$$PI_k = I \quad (k = 1)$$

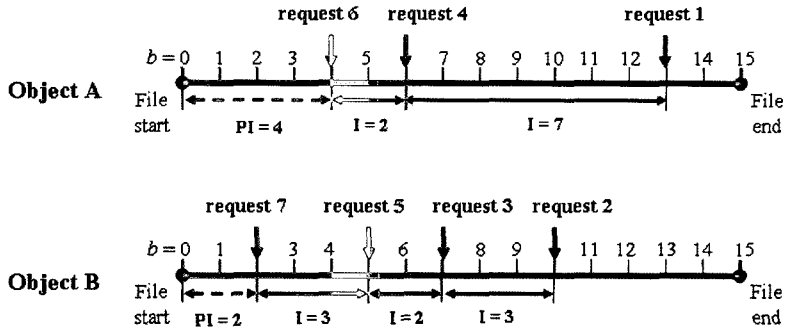


그림 2 이득값 계산. 객체 A에 대한 기대 참조 확률 pA = 1/PI = 0.25이고, 시작 부분 블록에 대한 이득값은 Profit(0) = 0.25/1, Profit(1) = 0.25/2, Profit(2) = 0.25/3, Profit(3) = 0.25/4이다. 이와 비슷하게 실제 인터벌에 속해 있는 블록들의 이득값은 Profit(4) = 1/1, Profit(5) = 1/2, Profit(6) = 1/1, Profit(7) = 1/2, Profit(8) = 1/3, Profit(9) = 1/4 이다. 캐싱 순서는 이득값에 의해 결정된다.

이 식에서 α는 0과 1사이의 상수로서 가장 최근의 실제 인터벌과 이전의 인터벌 간의 비중을 결정한다. B-PIC 기법에서는 실험적 분석을 통해 α의 기본값을 0.6으로 정했다. 수식 (2)는 수식 (1)에서 구한 PI 값을 바탕으로 멀티미디어 객체 i의 기대 참조 확률 p_i를 구하는 식이다. PI 값이 크면 가까운 미래에 해당 멀티미디어 객체가 요청될 확률이 작음을 뜻하고, 반대로 PI 값이 작으면 가까운 미래에 요청될 확률이 크다는 것을 뜻한다.

$$p_i = 1 / PI \quad (2)$$

마지막으로, 기대 참조 확률을 바탕으로 멀티미디어 객체 i의 모든 블록에 대해 이득값(profit)을 계산한다. 각 블록의 이득값은 해당 블록을 캐싱했을 때 얻을 수 있는 캐쉬 적중률을 뜻한다. 즉, 이득값이 큰 블록은 캐쉬에서 적중될 확률이 크고, 반대로 이득값이 작은 블록은 캐쉬에서 적중될 확률이 작다. 따라서, 이득값이 큰 블록을 캐싱하는 것이 시스템의 성능 면에서 효과적이다. 각 블록에 대한 이득값 계산은 가상 인터벌에 속한 시작 부분 블록과 실제 인터벌에 속한 블록에 대해 따로 나누어서 구한다. 시작 부분 블록(prefix block)의 이득값(profit)은 다음과 같이 계산된다.

$$Profit_i(b) = p_i \cdot 1 / t_b \quad (3)$$

수식 (3)에서 t_b는 객체 i가 현재 시점에 참조될 것이라는 가정 하에 블록 b가 참조될 때까지 걸리는 시간을 나타낸다. 여기서의 시간은 하나의 블록이 참조될 때마다 1씩 증가하는 논리적인 시간을 뜻한다. 예를 들어, 객체 i가 현 시점에 참조되기 시작할 때 처음 3개의 블록의 t_b 값은 각각 1, 2, 3이 된다. 이와 비슷하게, 실제 인터벌 i에 속한 각각의 블록 b의 이득값은 수식 (4)와 같이 계산한다.

$$Profit_i(b) = 1 / t_b \quad (4)$$

여기서 t_b는 블록 b가 참조될 때까지의 걸리는 시간을 나타낸다. 실제 인터벌에 대한 기대 참조 확률 p_i의 값은 1이기 때문에 수식 (4)에서는 p_i 값을 제거하였다. B-PIC 기법은 모든 블록에 대한 이득값을 계산하고, 이득값이 높은 블록부터 캐싱한다. 그림 2는 블록에 대한 이득값을 계산하는 예를 보여주고 있다. 특히, 그림 2에서 빈번하게 요청되는 객체 B의 PI 값은 상대적으로 객체 A의 PI 값과 실제 인터벌보다 작아 시작 부분의 블록을 캐싱할 가능성이 크다. 즉 인기도가 높은 객체의 시작 부분을 캐싱할 수 있다.

한편, 기존의 PIC 기법에서는 시간이 흐름에 따라 실제 요청과 가상 요청 사이에 성립되었던 가상 인터벌의 원도우도 같이 멀티미디어 객체의 뒷부분으로 이동한다. 그 결과 시작 지연 시간(startup latency)을 효과적으로 줄일 수 있는 멀티미디어 객체의 앞부분(prefix of object)을 캐싱하고 있지 못할 수 있다. 그림 3은 이러한 문제점을 설명하고 있다. 그와 반대로, 본 논문이 제안한 B-PIC 기법에서는 시간이 흘러도 메모리 버퍼에 인기 있는 객체의 시작 부분 블록을 캐싱하고 있어서 인기 있는 객체의 초기 서비스 지연 시간을 줄일 수 있다. 결국 B-PIC 기법은 PIC 기법을 블록 수준에서 재정립함으로써, 캐싱의 이득을 극대화시킬 수 있다. 또한, 시작 부분 캐싱 개념을 통해서 실제 요청되기 전에 인기 있는 객체의 시작 부분을 캐싱하기 때문에, 사용자가 기다려야 하는 초기 지연 시간을 줄일 수 있다.

3.3 알고리즘의 효율적인 구현 방법

본 절에서는 B-PIC 알고리즘에 대한 효율적인 구현 방법을 설명한다. 3.2절에서 설계한 알고리즘에 의하면 블록의 이득값은 시간이 지남에 따라 값이 변하게 되므

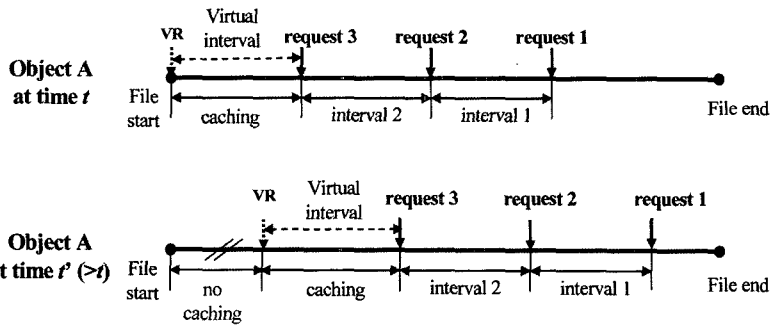


그림 3 시간 t일 때, PIC 기법은 객체의 시작부분을 포함하는 가상 인터벌(virtual interval)을 캐싱한다. 하지만, 시간이 지나면서, 가상 인터벌의 윈도우가 이동하고 가상 인터벌은 시작 부분을 포함하지 않는다. 그 결과 객체 A의 시작부분은 캐싱되지 않고, t 시간 이후에 객체 A에 대한 요청이 들어오면 객체의 시작 부분은 캐쉬 미스(cache miss)될 것이다.

로 매번 새로운 인터벌에 대한 캐싱 여부를 결정하고 기존 블록에 대한 교체가 필요할 때마다 모든 블록의 이득값을 재계산해야 하는 과정이 필요할 수 있다. 이러한 방법은 캐쉬 블록의 수가 n개라고 할 때 알고리즘의 시간 복잡도를 $O(n)$에 이르게 하여 온라인으로 캐쉬 운영을 하는 것을 현실적으로 어렵게 한다. 하지만, 다음 성질은 이러한 시간 복잡도를 해결하여 효율적인 캐쉬 운영을 가능하게 한다.

성질 1. 임의의 캐쉬 블록 a, b에 대해 시각 t에서 $Profit_t(a) > Profit_t(b)$ 이면 $t' > t$ 인 임의의 시각 t' 에 대해 $Profit_{t'}(a) > Profit_{t'}(b)$ 를 만족한다. □

(증명) $Profit(a) = 1/p$, $Profit(b) = 1/q$ 일 때,
시각 t 에 $p < q$ 이므로 $Profit_t(a) > Profit_t(b)$이다.
또한, 시각 t' 에 $t' > t$ 이고 $t'-t = \delta$ 라 하면
$p-\delta < q-\delta$이므로, 이때도 $Profit_{t'}(a) > Profit_{t'}(b)$ 이다.

성질 1은 캐쉬 내에 있는 블록 중에서 참조되지 않은 블록들 간에는 이득값의 대소 관계가 그대로 유지된다는 의미로서, 힙(heap)구조를 이용하여 시간 복잡도 $O(\log_2 n)$에 캐쉬 운영을 할 수 있는 바탕이 된다. 그림 4는 이러한 성질을 바탕으로 각 객체들을 이득값의 순서에 따라 최소 힙(min. heap) 구조로 유지하는 모양을 나타낸 것이다. 성질 1에 의해 시간이 흐르면 이득값 자체는 변하지만 이득값의 대소 관계가 변하지 않으므로 힙 구조는 그대로 유지된다. 새로운 인터벌이 형성되는 경우 힙의 루트에 있는 블록만 현재 시각에서의 이득값을 재계산한 후 캐싱을 추가적으로 수행할지 아닐지를 결정하면 된다. 이 때 새롭게 캐싱을 하기로 결정되었다면 힙의 루트에 있는 블록을 삭제한 후 새로운 블록의 이득값에 맞는 힙 상의 위치를 찾아주면 된다. 이는 성질 1을 이용하여 위치를 찾는 경로 상에 있는 노드들만

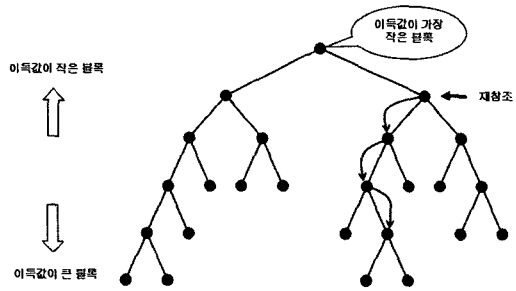


그림 4 힙(Heap) 구조에 기반한 캐쉬 운영

현재 시점에서의 이득값을 재계산하면 되므로 이러한 연산은 모두 $O(\log_2 n)$에 수행할 수 있다.

알고리즘의 공간 복잡도는 각 블록의 참조확률만 유지하면 되므로 블록당 $O(1)$의 공간으로 해결할 수 있다.

4. 성능 평가

본 장에서는 B-PIC 기법의 성능을 기존의 알고리즘과 비교한다. 각 알고리즘의 성능 비교를 위해 트레이스 기반 시뮬레이션을 통해 실험하였다. 또한 효율적인 실험을 위해서 두 곳의 상용 VOD 사이트(OnGameNet, Hanmir)로부터 추출한 실제 사용자 요청 트레이스를 사용하였다[13,14]. OnGameNet 트레이스는 293개의 비디오 파일을 가지고 있으며, 이들 비디오 파일의 평균 실행 시간은 883초이다. 비디오 파일에 대한 평균 요청 간격(inter-arrival time)은 6초이다. Hanmir 트레이스는 1266개의 비디오 파일을 가지고 있으며, 평균 실행 시간은 1078초이다. 비디오 파일에 대한 평균 요청 간격은 21초이다. 표 1은 이들 트레이스의 특성을 정리한 것이다. B-PIC과의 비교 대상으로는 PIC(Popularity-aware Interval Caching), IC(Interval Caching), LRU

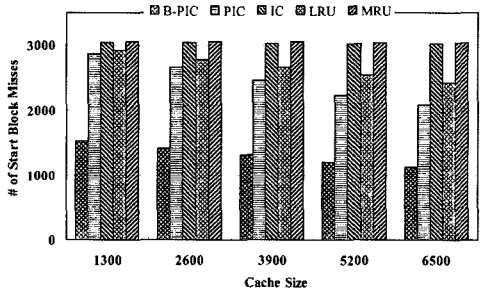
표 1 각 트레이스 별 특성

	OnGameNet trace	Hanmir trace
요청 수	3091	6938
평균 실행 시간	883 초	1078 초
평균 요청 간격	6 초	21 초
객체 수	293	1266

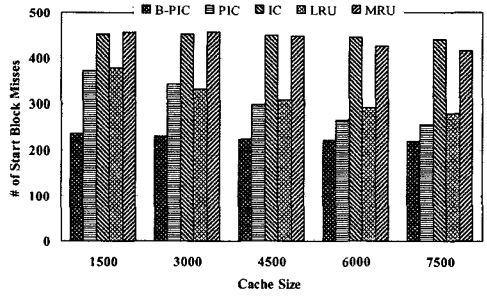
(Least Recently Used), MRU(Most Recently Used) 알고리즘 등을 사용했다.

그림 5는 캐쉬의 크기 변화에 따른 각 캐싱 기법의 시작 블록 미스 횟수(number of start block misses)를 비교하고 있다. B-PIC 기법은 실제 요청이 발생하기 전에 인기 있는 멀티미디어 객체의 시작 부분을 캐싱하고 있기 때문에 다른 캐싱 기법보다 훨씬 좋은 성능을 보이고 있다. PIC 기법도 실제 요청이 발생하기 전에 예측된 인터벌을 통해 객체의 시작 부분을 캐싱하기 때문에 인터벌 캐싱, LRU, MRU 기법보다는 좋은 성능을 보였다. 하지만, PIC 기법에서는 시간이 흐름에 따라 예측된 인터벌 구간이 점차 뒷부분으로 이동하기 때문에 B-PIC 기법보다 안 좋은 성능을 나타내었다. B-PIC 기법은 인기 있는 멀티미디어 데이타의 시작 부분을 캐싱하기 때문에 시작 블록 미스 횟수에 대해서 기존의 PIC 기법보다 46.7%의 성능 향상을 보일 수 있었다.

그림 6은 캐쉬 크기의 증가에 따른 각 캐싱 기법의 캐쉬 적중률(hit ratio)을 보여주고 있다. 1장에서 언급한 것과 마찬 가지로 전통적인 버퍼 캐쉬 관리 기법인 LRU와 MRU는 본 실험에서 우수한 성능을 나타내지 못하였다. 본 논문이 제안한 B-PIC 기법은 두 개의 트레이스를 사용한 실험에서 모두 기존 캐싱 기법들에 비해 가장 우수한 성능을 나타내었다. 인터벌 캐싱 기법도 B-PIC 기법보다는 떨어지지만 다른 캐싱 기법보다 좋은 성능을 보이고 있다. PIC 기법은 OnGameNet 트레이스에서는 인터벌 캐싱 기법과 비슷한 성능을 나타내었지만, Hanmir 트레이스에서는 B-PIC과 인터벌 캐싱 기법보다 낮은 성능을 나타내었다. 이는 Hanmir 트레이스의 요청 간격이 상대적으로 길고 그 편차 또한 크기 때문으로 분석할 수 있다. 이와 같은 특성으로 인해 PIC 기법에서는 예측 인터벌에 대한 정확성이 급격히 떨어져 좋지 않은 성능을 나타낸 것이다. 하지만 B-PIC 기법의 경우, 기대 참조 확률의 개념에 근거해 블록의 이익을 계산하기 때문에 실제 인터벌에 속한 블록의 우선순위가 상대적으로 더 높아 이와 같은 현상을 극복할 수 있었다.

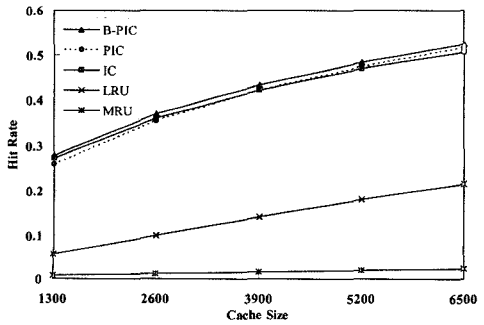


(a) OnGameNet 트레이스

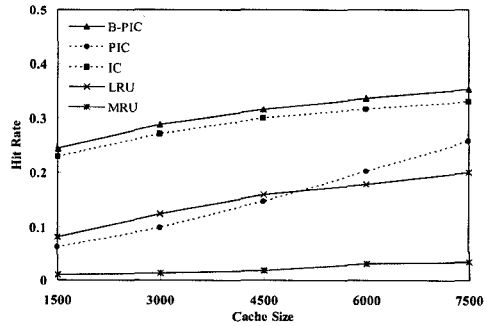


(b) Hanmir 트레이스

그림 5 B-PIC, PIC, IC, LRU, MRU 기법 별 시작 블록 미스 횟수 비교



(a) OnGameNet 트레이스



(b) Hanmir 트레이스

그림 6 B-PIC, PIC, IC, LRU, MRU 기법 별 캐쉬 적중률 비교

5. 결 론

본 논문에서는 멀티미디어 스트리밍 서버를 위한 인기도 기반 인터벌 캐싱의 블록 수준 세분화 기법을 제안하였다. 새롭게 제안된 기법은 인터벌을 구성하는 일련의 블록 단위로 캐싱의 우선 순위를 계산하는 기존의 기법들과 달리 인터벌을 구성하는 각각의 블록 단위로 캐싱의 우선 순위를 결정하여 더욱 높은 성능을 나타낼 수 있었다. 각 블록의 가치를 평가하는데 있어서는 기대 참조 확률의 개념을 사용하였으며, 실제 인터벌 뿐 아니라 인기 있는 멀티미디어 객체의 시작 부분을 캐싱하기 때문에, 기존의 PIC, IC, LRU, MRU보다 캐쉬 적중률 및 시작 부분 미스에 대해서 우수한 성능을 나타내었다. 특히, 시작 블록 미스에 관해서는 기존의 PIC 기법보다 46.7%의 성능 향상을 나타내었다.

참 고 문 헌

[1] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report RC19347, T.J. Watson Research Center, Yorktown Heights, NY.

[2] A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments," Proceedings of SPIE Multimedia Computing and Networking Conference, San Jose, CA, 1996.

[3] N. J. Sarhan and C. R. Das, "Caching and Scheduling in NAD-Based Multimedia Servers," IEEE Transactions on Parallel and Distributed Systems, Vol.15, No.10, pp.921-933, Oct. 2004.

[4] J. M. Almeida, D. L. Eager, M. K. Vernon, "A Hybrid Caching Strategy for Streaming Media Files," Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking, 2001.

[5] B. Ozden, R. Rastogi and A. Silberschatz, "Buffer Replacement Algorithms for Multimedia Storage Systems," Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pp. 172-180, 1996.

[6] B. Ozden, R. Rastogi and A. Silberschatz, "Disk Striping in Video Server Environments," Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pp. 580-589, 1996.

[7] T. Kim, H. Bahn, and K. Koh, "Popularity-Aware Interval Caching for Multimedia Streaming Servers," IEE Electronics Letters, Vol.39, No.21, pp. 1555-1557, Oct. 2003.

[8] T. Kim, H. Bahn, and K. Koh, "Efficient Cache Management for QoS Adaptive Multimedia Streaming Services," Lecture Notes in Computer Science, Springer-Verlag, Vol.3768, pp.1-11, Oct. 2005.

[9] K. Lee, Y. Y. Park, H. Y. Yeom, "Pre-emptive but safe interval caching for real-time multimedia system," Int'l Journal of Computer Systems Science and Engineering, Vol. 18, No. 2, pp. 87-94, 2003.

[10] J. Fernandez, J. Carretero, F. Garcia-Carballeira, A. Calderon, and J. Perez-Menor, "New stream caching schemas for multimedia systems," IEEE Int'l Conf. Automated Production of Cross Media Content for Multi-Channel Dist., 2005.

[11] Sen, S., Rexford, J., and Towsley, D., "Proxy prefix caching for multimedia streams," IEEE INFOCOM'99, 1999.

[12] Eun-Ji Lim, Seong-Ho Park, Hyeon-Ok Hong, and Ki-Dong Chung, "A proxy caching scheme for continuous media streams on the Internet," 15th International Conference on Information Networking, pp. 720-725, 2001.

[13] OnGameNet Co. Ltd, http://www.ongamenet.com.

[14] Hanmir, Co. Ltd, http://www.hanmir.net.

권 오 훈

2002년 세종대학교 전산과학과 이학사 2004년 서울대학교 컴퓨터공학부 공학석사. 2004년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 멀티미디어 캐싱 시스템, 운영체제

김 태 석

2000년 서울대학교 전산과학과 이학사 2002년 서울대학교 컴퓨터공학부 공학석사. 2002년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 멀티미디어 파일시스템, QoS 시스템, 운영체제

반 효 경

정보과학회논문지 : 시스템 및 이론 제 34 권 제 2 호 참조

고 건

1974년 서울대학교 응용물리학과 이학사 1979년 버지니아대학교 전산학과 공학석사. 1981년 버지니아대학교 전산학과 공학박사. 1983년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 멀티미디어 QoS 시스템, 운영체제